phpSAFE: A Security Analysis Tool for OOP Web Application Plugins

Paulo Nunes, José Fonseca
CISUC, University of Coimbra
UDI, Polytechnic Institute of Guarda
Portugal
pnunes@ipg.pt, josefonseca@ipg.pt

Marco Vieira
CISUC, University of Coimbra
Portugal
mvieira@dei.uc.pt

Abstract—**There is nowadays an increasing pressure to develop complex web applications at a fast pace. The vast majority is built using frameworks based on third-party server-side plugins that allow developers to easily add new features. However, as many plugin developers have limited programming skills, there is a spread of security vulnerabilities related to their use. Best practices advise the use of systematic code review for assure security, but free tools do not support OOP, which is how most web applications are currently developed. To address this problem we propose phpSAFE, a static code analyzer that identifies vulnerabilities in PHP plugins developed using OOP. We evaluate phpSAFE against two well-known tools using 35 plugins for a widely used CMS. Results show that phpSAFE clearly outperforms other tools, and that plugins are being shipped with a considerable number of vulnerabilities, which tends to increase over time.**

Keywords—Static analysis; web application plugins; security; vulnerabilities

I. Introduction

The number and importance of web applications is growing exponentially fast in a world where web browsers are ubiquitous and used by everyone. Nowadays, almost everything is available, displayed, discussed, shared, processed or traded on the web and the demand for building new web applications is enormous. In fact, millions of people rely on web applications, either for work or leisure, accessing and managing sensitive information like financial and personal data.

The demand for developing web applications with increased complexity in very tight time constraints is making a huge pressure to build more at a lower cost. Many web applications are built on top of Content Management Systems (CMS) frameworks that can be easily deployed and customized to meet the requirements of a myriad of different scenarios, like personal web sites, blogs, social networks, webmail, banking, e-commerce, etc. To cope with this diversity, most CMS-based applications can be extended and configured with third-party server-side plugins provided by multiple developers.

The number of CMS applications is huge and there are thousands of plugins provided by third-party developers. In practice, developers with unknown agendas and uncontrolled software programming skills implement these plugins, which ultimately leads to suspicious trust levels. The problem becomes even worse as core CMS providers do not run quality assurance procedures on the third-party plugins they use, even for those made available directly on untrusted web sites. The only guarantee is related with the comments and ratings of the plugins by their end users and the number of downloads. This is clearly not enough and the reality is that no one can fully trust on the security of the plugins. Not surprisingly, many of these plugins can be exploited by malicious minds, which is the case in many situations [1].

Source code review is a key activity to improve the security of a software product. This can be confirmed by its relevance in the most important secure software development lifecycles, like Microsoft SDL, CLASP and Software Security Touchpoints [2]. However, source code review is a resource intensive task that is only feasible if supported by automated tools. There are several tools that can be used, but the vast majority of plugin developers cannot afford the expensive commercial source code analyzers. Although they can use free tools, like RIPS or Pixy, a key limitation of these tools is the absence of capabilities to analyze Object Oriented Programming (OOP) code, which is nowadays largely used for developing CMS applications [3]. Another drawback is the lack of knowledge on the CMS framework for which the plugin is being developed. Indeed, when analyzing the plugin, such tools are not aware of input and output vectors and of filtering functions included in the API of the CMS framework. These limitations lead to vulnerabilities being left undetected and at the same time to the generation of many false alarms.

In this paper we present phpSAFE, a source code analyzer for PHP based plugins able to detect Cross Site Scripting (XSS) and SQL Injection (SQLi) vulnerabilities. phpSAFE is a follow-up of a project whose development was requested by Automattic, the developer of WordPress [4], with the goal of improving the security of a number of plugins. The tool was developed from the ground up with OOP and plugin security in mind, thus including OOP concepts like objects, properties and methods. To evaluate phpSAFE we compared its ability to detect plugin vulnerabilities with two well-known free tools, RIPS and Pixy. As target CMS framework we selected WordPress because it accounts for about 23% of the all the web sites [4] and has a market share of approximately 60% among all CMS [5]. From the vast collection of plugins available we selected 35 with different sizes and complexities. As we also wanted to study how the tools behave regarding the evolution of plugins over time, we considered two versions of each plugin: one from 2012 and another from 2014.

Results show that phphSAFE is able to detect more vulnerabilities than the other tools, with fewer false alarms. We also observed that both phpSAFE and RIPS deal well with the evolution of plugin code. A key observation is that plugins that are currently being used in thousands of WordPress installations have dangerous XSS and SQLi vulnerabilities and this number is increasing over the years. In fact, we discovered more than 580 vulnerabilities in the plugins analyzed, many of them very easy to detect and exploit. We also verified that developers did not fix many vulnerabilities even after knowing them for more than

PEst-OE/EGE/UI4056/2014 – project financed by Science and Technology Foundation
Project ICIS - Intelligent Computing in the Internet of Services (CENTRO-07-ST24-FEDER-002003), co-financed by QREN, in the scope of the Mais Centro Program and European Union's FEDER